图文 83 基于订单定时退款场景，来分析RocketMQ的延迟消息的代码实现

202 人次阅读 2020-01-21 09:16:49

详情 评论

基于订单定时退款场景，来分析RocketMQ的延迟消息的代码实现

继《从零开始带你成为JVM实战高手》后，救火队长携新作再度出山，重磅推荐：

（点击下方蓝字试听）

《从零开始带你成为MySQL实战优化高手》

上一篇文章我们分析了延迟消息的使用场景，这篇文章我们分析一下RocketMQ中对延迟消息的代码实现

其实RocketMQ对延迟消息的支持是很好的，实现起来也非常的容易，我们先看发送延迟消息的代码示例。

```
public class ScheduledMessageProducer {

  public static void main(String[] args) throws Exception {
    // 这是订单系统的生产者
    DefaultMQProducer producer =
        new DefaultMQProducer("OrderSystemProducerGroup");
    // 启动生产者
    producer.start();

    Message message = new Message(
       "CreateOrderInformTopic",  // 这是创建订单通知Topic
       orderInfoJSON.getBytes() // 这是订单信息的json串
    );
    // 这里设置了消息为延迟消息，延迟级别为3
    message.setDelayTimeLevel(3);

    // 发送消息
    producer.send(message);
  }

}
```

大家看上面的代码，其实发送延迟消息的核心，就是设置消息的**delayTimeLevel**，也就是延迟级别

RocketMQ默认支持一些延迟级别如下：1s 5s 10s 30s 1m 2m 3m 4m 5m 6m 7m 8m 9m 10m 20m 30m 1h 2h

所以上面代码中设置延迟级别为3，意思就是延迟10s，你发送出去的消息，会过10s被消费者获取到。那么如果是订单延迟扫描场景，可以设置延迟级别为16，也就是对应上面的30分钟。

接着我们看看一个消费者的代码示例，比如订单扫描服务，正常他会对每个订单创建的消息，在30分钟以后才获取到，然后去查询订单状态，判断如果是未支付的订单，就自动关闭这个订单

```
public class ScheduledMessageConsumer {

   public static void main(String[] args) throws Exception {
      // 订单扫描服务的消费者
      DefaultMQPushConsumer consumer =
          new DefaultMQPushConsumer("OrderScanServiceConsumer");
      // 订阅订单创建通知Topic
      consumer.subscribe(""CreateOrderInformTopic", "*");
      // 注册消息监听者
      consumer.registerMessageListener(new MessageListenerConcurrently() {
          @Override
          public ConsumeConcurrentlyStatus consumeMessage(
              List<MessageExt> messages, ConsumeConcurrentlyContext context) {
              for (MessageExt message : messages) {
                // 这里打印一下消息的存储时间到消费时间的差值
                // 大概就是我们设置的延迟级别的时间
                System.out.println("Receive message[msgId=" +
                                   message.getMsgId() + "] " +
                                   (System.currentTimeMillis() - message.getStoreTimestamp()) +
                                   "ms later");
              }
              return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
          }
      });
      // 启动消费者
      consumer.start();
}}
```

把延迟消息的使用搞明白了之后，想必大家以后在自己的系统中就可以使用延迟消息去支持一些特殊的业务场景了。

End

狸猫技术窝精品专栏及课程推荐：

《从零开始带你成为JVM实战高手》
《21天互联网Java进阶面试训练营》（分布式篇）
《互联网Java工程师面试突击》（第1季）
《互联网Java工程师面试突击》（第3季）

重要说明：

如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑

如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

具体加群方式，请参见目录菜单下的文档：《付费用户如何加群》（**购买后可见**）